Point Set Surface Reconstruction for VTK

Release 0.00

David Doria

January 31, 2010

Rensselaer Polytechnic Institute, Troy NY

Abstract

This document presents a set of classes (*vtkPointSetSurfaceReconstruction*, *vtkVoxelizePolyData*) to produce a surface from an oriented point set. These classes are implemented as VTK filters. A Paraview plugin interface is provided to allow extremely easy experimentation with the new functionality. We propose these classes as an addition to the Visualization Toolkit.

Latest version available at the Insight Journal [http://hdl.handle.net/10380/3149]

Contents

1 Introduction

There are several data acquisition methods which produce 3D points as output. Examples include Light Detection and Ranging (LiDAR) scanners, Structure From Motion (SFM) algorithms, and Multi View Stereo (MVS) algorithms. It is often necessary to produce a mesh from these points. This paper presents a simple algorithm for producing a surface from a set of 3D points which have an associated normal vector.

2 vtkVoxelizePolyData

vtkVoxelizePolyData is a supporting class for this work. Its only function is to overlay a grid of voxels onto the input point set. This grid is necessary for the main algorithm, which is presented in the next section.

3 vtkPointSetSurfaceReconstruction

To estimate a surface from a set of oriented points, we first produce a discretely sampled volume over the points using vtkVoxelizePolyData filter, described above. For each point p_g in this grid, we then find the closest point in the input point set, p_c. We then compute the signed distance from p_g to the plane defined by p_c and the normal vector associated with p_c. This value is stored as scalar data at p_g. The zero level set of this volume is then extracted using vtkContourFilter. The idea is that surface we are seeking is the set of all points where the distance from the point to the plane defined by the nearest input point is zero.

3.1 Options

SamplesPerDimension

This variable sets the density of the volumetric grid. The higher this value, the smoother the resulting mesh will be.

This value can be set using:

```
surfaceReconstructionFilter->SetSamplesPerDimension(100);
```

Border

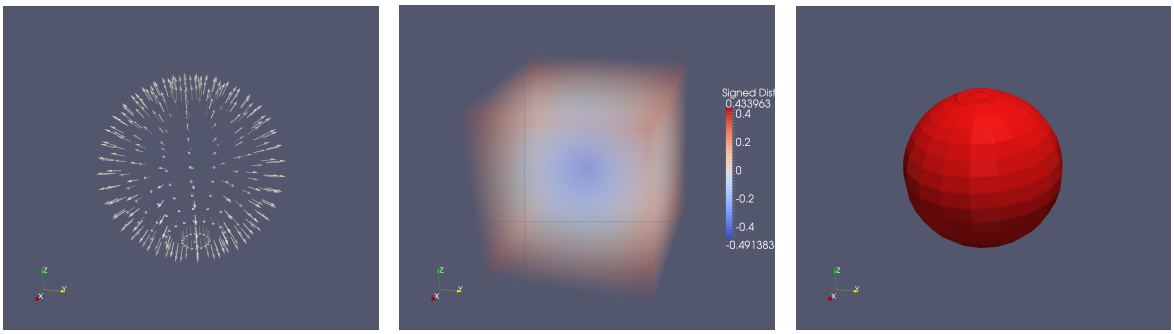
This variable sets how far outside the bounds of the input data the grid stretches. We have observed some strange “edge effects” that can be remedied by changing this value. The correct value seems to be a function of the size of the cells.

This value can be set using:

```
surfaceReconstructionFilter->SetBorder(5);
```

3.2 Demonstration

To demonstrate surface reconstruction, we use points sampled from a sphere. We computed the normals of these points using a previously submitted class, vtkPointSetNormalEstimation. We oriented these normals using another previously submitted class, vtkPointSetNormalOrientation. The resulting oriented point set is shown in Figure 1(a). The signed distance volume, an intermediate step, is shown for clarity in 1(b). In Figure 1(c), we show the final reconstructed surface.



(a) Oriented point set

(b) Interpolated volume of signed distances

(c) Sphere surface

Figure 1: Surface reconstruction demonstration.

3.3 Code Snippet

```
vtkSmartPointer<vtkPointSetSurfaceReconstruction> surfaceReconstructionFilter =
vtkSmartPointer<vtkPointSetSurfaceReconstruction>::New();
surfaceReconstructionFilter->SetInput(inputPolyData);
surfaceReconstructionFilter->SetSamplesPerDimension(100);
surfaceReconstructionFilter->SetBorder(5);
surfaceReconstructionFilter->Update();

vtkPolyData* surface = surfaceReconstructionFilter->GetOutput();
```

4 Future Work

The algorithm implemented in this paper is very sensitive to noise error in the normal computation, the grid spacing, and the density of the points. There are several newer techniques, such as Poisson surface reconstruction, that we intend to implement for VTK in the future.

5 Paraview Plugin

For convenience, this code is shipped with a Paraview filter plugin. The plugin provides an easy way to set the parameters as well as integrate the new code into your workflow. A screenshot of the plugin interface is shown in Figure 5.

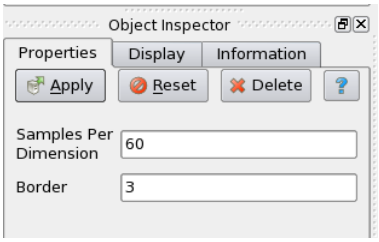


Figure 2: Paraview plugin screenshot